

Orchestrating Computing Objects Using Web-Services Technologies

Intel IT Proof of Concept (POC)

Guy Yohanan, Jay Hahn-Steichen, Jim Hobbs, Jackson He
Intel Corporation
{guy.yohanan, jim.hobbs, jay.hahn-steichen, jackson.he}@intel.com

Abstract

The challenge of ever-growing complexity within the IT enterprise accelerates the evolution of a fully autonomic framework. With the need of an agile IT infrastructure for ever-changing business needs, this evolution is bound to develop based on open standards. We have conducted a proof of concept (POC) using web services (WS) technologies in an enterprise manageability framework. The proof of concept focuses on a standard messaging layer for enterprise integration, standard and discoverable manageability interfaces, and a holistic object management approach. This paper presents a POC that uses WS standards to build a common “language” between varieties of components within the enterprise. That common “language” includes discoverable interfaces and a local metadata repository. Having this common communications framework with the ability to discover and control computing resources moves us ahead significantly toward the vision of an agile IT infrastructure, and enables the continuous evolution toward the vision of an autonomic enterprise in which holistic, end-to-end object management is possible. WS-Management (WS-Man), which is a messaging protocol dedicated to the purpose of system management, is the primary specification used in the POC. Among other things, this paper presents models to deal with key challenges like dynamic managed-object (MO) discovery and control, federated event processing across domains, and systems which reach beyond technology into business layers. The success of this POC proved that such an end-state vision can be achieved, and is a major step forward toward this goal.

1. Introduction

The stack integration challenge is an ever-growing issue within the enterprise. In the enterprise manageability framework, complexity continues to grow exponentially mainly due to:

- Multiple and disparate managed object (MO) types:
 - Platforms
 - Hardware
 - Operating systems
 - Applications
- Multiple management toolsets
- Implementation of IT Infrastructure Library (ITIL) Functions
- Specific communication protocols for MO’s, management tools, and ITIL functions, which are often proprietary
- Proprietary manageability interfaces (or in some cases, no interface at all)
- And most critically – all are non-discoverable

Those challenges are true barriers to the evolution of the “autonomic enterprise” and the ability of an IT shop to be agile in response to changing business needs.

1.1. The Proof of concept project

IT shops are interested in a simple, integrated and self-contained enterprise in which IT services can be delivered with higher quality and lower total cost of ownership (TCO). In order to achieve this goal, all components within the enterprise need to “talk” the same language, and must be able to be discovered and freely communicate with each other.

To that purpose, we conducted a proof of concept (POC) using web services (WS) technologies in an enterprise manageability framework. The proof of concept focused on these areas:

- Standard messaging layer for enterprise integration
- Standard and discoverable manageability interfaces
- A holistic approach to object management

The POC used WS standard specifications in the enterprise manageability framework to build a loosely-coupled standard messaging layer as “glue” between the different components in the enterprise, using standard and discoverable manageability interfaces and a local metadata repository. This enabled a holistic, end-to-end approach to object management within the enterprise, simplified stack integration and enterprise manageability, and promoted the autonomic enterprise vision. The POC explored automated solutions to real IT problems that required additional layers built on top of this foundation.

1.2. Web Services Technologies

Web services (WS) technologies employ platform-independent standards based on XML to communicate within distributed systems. They allow loosely coupled, short-term cooperation between services. The basic protocol defining this type of communication between web services is SOAP (Simple Object Access Protocol). [WS-Management](#) (WS-Man) is a messaging protocol dedicated to system management that relies on multiple web service specifications to accomplish its tasks. WS-Man is a general SOAP-based protocol for managing systems such as PCs, servers, devices, and web services as well as applications, and other manageable objects. It defines a loosely-coupled, standard, reusable manageability interface, a metadata representation model, and a messaging layer for discovering, accessing and using the interface. The specification includes a standard eventing mechanism, subscription capabilities, resource manipulation, service cataloging, etc.

http://www.intel.com/technology/manage/download/s/ws_management_june_2005.htm

2. Orchestrating Computing Objects using Web-Services Technologies in Enterprise Manageability

2.1. Autonomic Enterprise and Agile IT Visions

The autonomic enterprise vision is that the computing capabilities that support an enterprise can function independently. They are well-integrated, and

can maintain themselves, and control their own resources to deliver IT services in a consistent and continuous way. The autonomic enterprise vision is part of an “agile IT” vision, allowing an IT shop to adjust itself all the time in a dynamic and automatic way, in response to changing business needs.

2.2. Creating an enterprise common “language”

As discussed above, one of the primary reasons for the complexity challenge is that the different components and systems in the enterprise use different protocols for communication, and are usually proprietary, resulting in immense integration efforts. A common “language” is needed to make enterprise components and systems discoverable and to enable them to freely communicate with each other. The POC demonstrates the use of the [WS-Management](#) (WS-Man) protocol stack, in front of each one of the stack layers, components and systems in the enterprise, in a reusable fashion. It enables a standard, common and unified messaging layer, and it drives MO discovery and provisioning. Together, these enable the development of a holistic approach to object management. Using a WS-Management protocol stack we were able to create a common communication approach for the entire enterprise manageability framework.

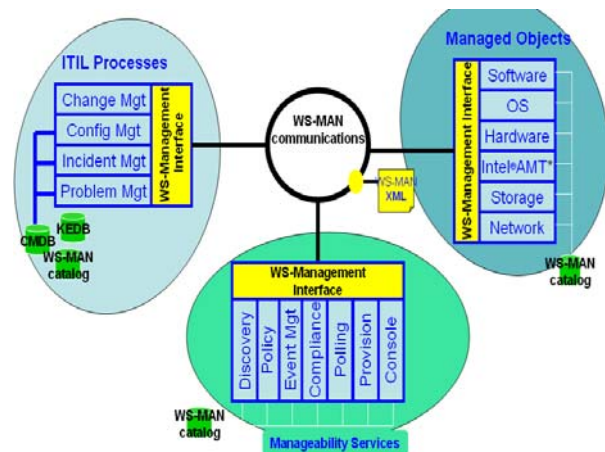


Figure 1: WS-Man stack in front of each component to create a common “language”

* Intel® Active Management Technology (Intel® AMT)

With such a common communications framework, enterprise manageability integration becomes much simpler, and provides the ability to discover and control computing resources. It moves us ahead significantly, and enables the continuous evolution of an autonomic enterprise as part of an agile IT vision.

2.3. Automated Discovery and Provisioning

A key enabler of the evolution of the autonomic enterprise is the ability to discover dynamically all computing resources and immediately control them.

Dynamic, real-time discovery means:

1. The ability to know when an MO is activated on the network or changes its state.
2. The ability to query the MO for its metadata.
3. Registration of the new MO instance in a Configuration Management Database (CMDB).

Immediate MO control is the ability to control the behavior of the object through business rules or policies. These are automatically provisioned to the MO when it is enabled or changes its state.

2.4. Dynamic, Real-time Managed Object Discovery

Conceptually, the dynamic, real-time MO discovery process has three main steps:

1. As the MO is activated on the network, it announces itself by sending a thin message.
2. The manager gets the announcement and queries the MO end-points (through the manageability interface) for the required object configuration item (CI) information.
3. The manager then registers the MO in the Configuration Management Database (CMDB).

As the MO joins the network, or has a metadata change, it sends an announcement – a thin “Hello” message, to a Discovery Proxy (DP) which is responsible for receiving those announcements and forwarding them to the manager (an enterprise level system). The manager then resolves the MO end-point (EP) URI (representing its manageability interface), which is provided in the thin “Hello” message. It then queries the MO for more information as needed for enterprise management. The MO Catalog (part of the WS-Man specification) contains the information needed by the manager to pull all required data, for example the events exposed by the MO, the methods available and how they can be used, or the MO properties and attributes. The manager then creates a new CI record in the CMDB for the new instance of the object type, or updates the record if it already exists (for example, a status change from “offline” to “online” or vice versa). The announcement process can be implemented through multicast or unicast. In the multicast case, a reference to the DP address could be part of the WS stack implementation. In the unicast

case, the DP address can be delivered as part of the initial provisioning of the system, or the MO EP could get the DP address when doing initialization (it could piggyback on DHCP). The DP will be equipped with a primary manager address and a secondary manager address. These are part of the DP’s policy information and are distributed to DP’s automatically. Then the DP will send a unicast message to the Manager with the newly discovered MO. The Manager can include an implementation of the DP.

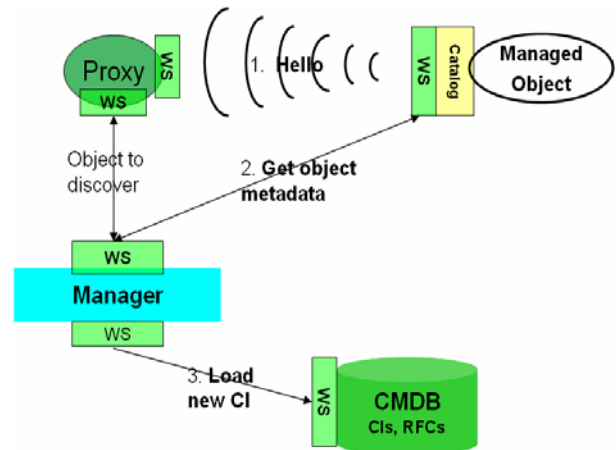


Figure 2: Managed-Object Discovery concept diagram

Dynamic, automatic, real-time managed object discovery is an essential capability for enterprise manageability as it evolves toward a fully autonomic framework. Discovery must address all layers of the stack, including hardware, the virtual machine monitor (VMM), operating systems, and applications. It is not limited to devices only. Discovery must be dynamic, automatic, and real-time (triggered when the MO changes state) in order to have an accurate picture of the state of the enterprise infrastructure and its services at any point in time so that the right IT business decisions can be made. Probing for a change (pull model) doesn’t answer enterprise management needs because current and future enterprise objects are highly dynamic and changing continually. The autonomic computing facilities will reconfigure themselves based on business need, load, fault management, and other factors in order to optimize service availability and throughput. Some key results of the discovery process are an up-to-date enterprise directory (CMDB, another key enabler of the evolution of an autonomic enterprise) and the functionality it enables. Linkages and associations between the MOs, provide the ability to distinguish between service management and infrastructure management, which is a critical element in managing and delivering IT services.

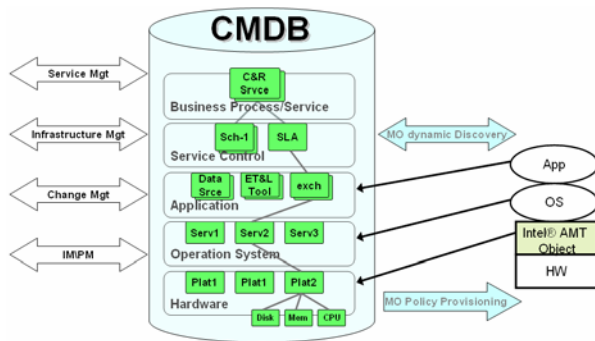


Figure 3: Managed-Object Discovery results

The specifications used in the POC to implement the discovery concept as illustrated above are [WS-Management](#) and [WS-Discovery](#). The latter used as a base-line specification for the thin “Hello” message (step #1 in the discovery process).

```

<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope
  xmlns:a="http://schemas.xmlsoap.org/ws/2004/08/addressing"
  xmlns:d="http://schemas.xmlsoap.org/ws/2004/10/discovery"
  xmlns:s="http://www.w3.org/2003/05/soap-envelope">
  <s:Header>
    <a:Action>http://schemas.xmlsoap.org/ws/2004/10/discovery/Hello</a:Action>
    <a:MessageID>uuid:3c571995-1f6f-42cc-9a1c-88fbbeaaf1c9</a:MessageID>
    <a:To>http://sysname01/DiscoveryWS/Discovery.asmx</a:To>
    <d:AppSequence InstanceId="11223344" MessageNumber="313" />
  </s:Header>
  <s:Body>
    <d:Hello>
      <a:EndPointReference>
        <a:Address>wsman:system/catalog/2005/02/Catalog</a:Address>
      </a:EndPointReference>
      <d:Types />
      <d:Sources>SERVER_NAME=SMAIDF01, CI_IMCLASS=SERVICE,CI_TYPE=Alert</d:Sources>
    </d:Hello>
  </s:Body>
</soap:Envelope>

```

Figure 4: “hello” message as MO Discovery step #1

The [WS-Management Catalog](#) is the key enabler for step #2 in the discovery process. The catalog is the entire set of metadata available from a WS-Management agent for a given resource. The catalog defines the manageability interface of the managed object, and the data model for its metadata. It allows dynamically querying the MO for the information and methods it exposes and defines how to use them in a true loosely-coupled fashion. [WS-Transfer](#), which is part of WS-Man specification suite, is then used to pull the information out of the catalog. The WS-Man standard communications protocol (SOAP/XML) is used for all communications between the different components participating in the discovery process described above.

```

<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/03/addressing"
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <wsa:Action>http://schemas.xmlsoap.org/ws/2004/09/transfer/GetResponse</wsa:Action>
    <wsa:MessageID>uuid:722d3f18-ad12-4e04-8710-cdb32fc05b62</wsa:MessageID>
    <wsa:RelatesTo>uuid:d5b7bff7-fe1a-4365-b3f6-8a0b86433e4b</wsa:RelatesTo>
    <wsa:To>http://schemas.xmlsoap.org/ws/2004/03/addressing/role/anonymous</wsa:To>
  </soap:Header>
  <soap:Body>
    <string><System Name="SYSNAME01" CatalogueUri="wsman:system/catalog/2005/02/Catalog"><Category Name="Hardware" CatalogueUri="wsman:intel.com/system/UnitaryComputerSystem"><Type Name="CPU" CatalogueUri="wsman:intel.com/system/Processor"><Instance Name="CPU0" /></Type><Type Name="DISK" CatalogueUri="wsman:intel.com/system/MediaAccessDevice"><Instance Name="\.\PHYSICALDRIVE0" /></Type><Type Name="MEMORY" CatalogueUri="wsman:intel.com/system/PhysicalMemory"><Instance Name="Physical Memory 0" /><Instance Name="Physical Memory 1" /></Type><Type Name="NETWORK"

```

Figure 5: MO end-point respond as part of MO Discovery step #2

2.5. Managed Object Policy Provisioning

Through the evolution of the autonomic enterprise, IT shops look to run their business based on a logical model that is descriptive of their business practices. This logical model is then mapped to physical business rules and policies. In the POC these physical policies were mapped to corresponding Managed-Object(MO) types, and physically provisioned to the MO.

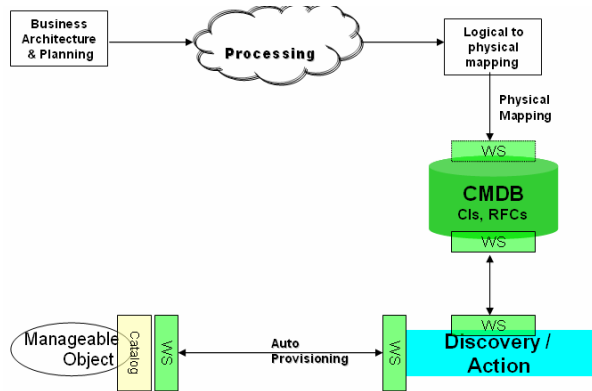


Figure 6: IT B'z Model rep and physical mapping (policies)

The concept we implement with this POC is that physical policies are stored in a CMDB, associated with the type of MO they reference. Once a new instance of an MO type is discovered, the MO attributes (like type, location, etc.) are used against selection criteria in the CMDB, to automatically trigger the provisioning of these policies. A compliance tool, once triggered, provisions the policies to the newly discovered MO through the standard WS-Man protocol and interface to a specific URI. The [WS-Transfer](#) specification, which is part of WS-Man suite, is then used to provision the policies to the MO, located at the specified URI.

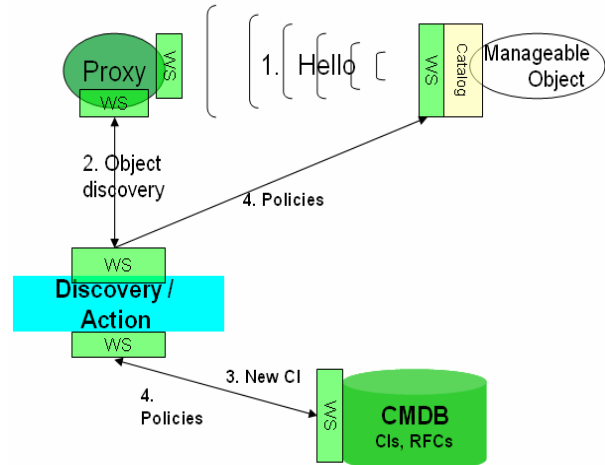


Figure 7: MO Discovery and Policies provisioning

2.6. The Role of Intel® Active Management Technology (Intel® AMT)

During the POC we used a simulator of Intel® AMT as part of a device's manageability stack. It had a major and foundational role, as it formed the root of the discovery and provisioning processes. When the device is first plugged into the network, the Intel® AMT simulator automatically and immediately comes online and is discovered (through the process described above). As part of this discovery, the Intel® AMT simulator sends its identification number as a unique identity key representing the host platform. This key serves as the root of the discovery and provisioning process for the entire stack, and announces the existence of the new device as a computing resource. As part of the POC we were able to automatically discover and provision a new server platform out-of-the-box by having the Intel® AMT simulator initiate the discovery process for the server "out-of-band" (OOB). In response to new server discovery, policies kept in the CMDB directed the bare-metal OS provisioning to be performed OOB via the Intel® AMT. Once the OS was brought online it was also discovered and the process of discovery and provisioning were repeated for the OS and for the entire stack (now in-band, or via a WS-man interface to the OS). Using this approach we were able to take a new hardware platform out of the box and automatically join it to an existing IT service for added computing power, with minimum delay and manual intervention.

2.7. Federated Event Processing

Enterprise event management can cross systems, domains, and company boundaries. It requires automated cooperation between multiple different enterprise systems, including business layer systems, to make it part of an automated response system. Those systems involved in processing an event include MO's, Management tool-set's, ITIL functions, and other systems like the Known Error Database (KEDB). The events MO's exposed are provided through the MO Catalog (WS-Man Catalog) and can be queried dynamically when needed. When a new instance of an object type is discovered, the event management system then triggered and made a subscription at the MO end-point for those events that are within the interest of the business (and those associated to the MO type within the CMDB as part of the physical model mapping). The subscription process can be done either directly by the event management system, or as part of the policy provisioning to that new object instance as it is discovered. [WS-Eventing](#) (part of WS-Man suite) provides the mechanism and protocol for event subscription.

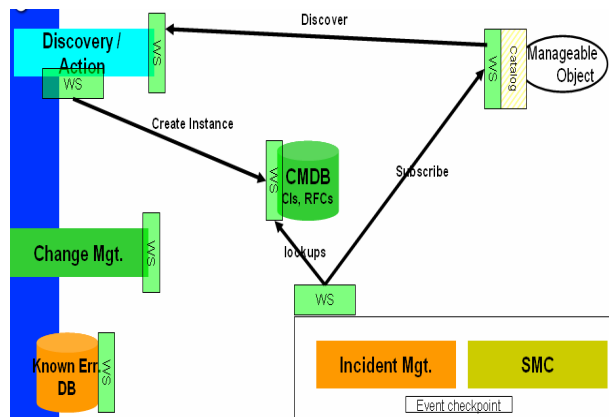


Figure 8: MO Discovery and Events Subscription

When a failure or fault event happened in an MO, the POC took a holistic approach to processing the failure that involved an integration of technical and IT business layers. In an example scenario, an event was sent to the event management system using [WS-Eventing](#), the event management system queried the CMDB for additional information needed with regards to the MO, it then processed the event internally. The event management system queried the Known Error Database (KEDB), using [WS-Transfer](#), for a potential known resolution for the problem. The KEDB replied with a resolution action to be executed (also using WS-Transfer). However it is not sufficient for computing

resources to make autonomic decisions in the enterprise. The business layer must be included to guarantee IT QoS. Therefore the event management system sent a request to the change management system (ITIL function at the business layer) for an approval and the timing for execution of the corrective action. The change management system replied with approval (both using WS-Transfer), and a request was sent to the compliance system (also using WS-Transfer) to execute the corrective action. The change authorization may include processing data like a window for maintenance, and expected time to perform. The compliance system then executes the corrective action using WS-Transfer which is part of the WS-Man specification suite. All the communications in the scenario above used the WS-Man standard protocol stack to provide standard communications, a unified messaging layer and the interface that enabled a holistic approach required to manage events in an automated fashion.

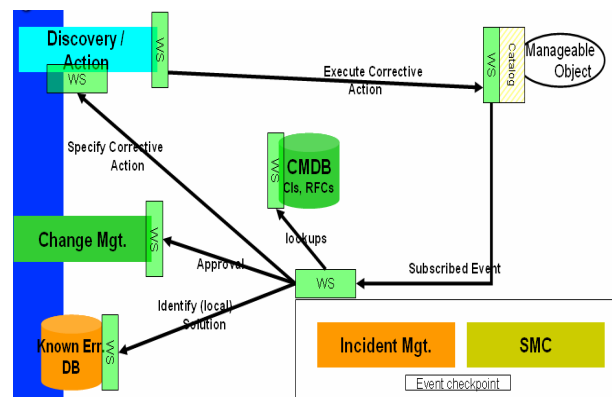


Figure 9: Federated Event Processing

2.8. Manageability Capabilities as Services in SOA

As we evolve toward service-oriented architecture (SOA), manageability capabilities will be exposed as services, and will enable standards-based boundary navigation. For example, the KEDB may be federated, with part of it physically located at a vendor site where it is exposed as another service to that vendor's customers. Events may be managed in part within an organization, and may also be sent directly to a vendor using a common WS specification. The decision of what events are managed where being a function of a policy specification.

3. Challenges

Some of the main challenges moving forward include the industry adoption of the WS-Management protocol suite as an open standard, the need for all layers in the stack, from HW to App, to “talk” WS-Man, and implement WS-Man standard and discoverable interfaces. IT shops need solutions that support WS-Man off the shelf. Another key challenge is around industry support for the complete MO discovery mechanism through standards, and to include discovery as part of the WS-Management specification. MO self-announcement is a key missing piece, most other parts of the discovery process can already be enabled via WS-Man. Extending Intel® AMT to all platforms as an enabler the root element of the device discovery process, and supports at hardware layer for the autonomic enterprise computing demonstrated in this paper. There is a need to promote a standardized data-model in the CMDB for MO’s with their attributes and associations, as well as a representation of policies and their association with MO types. Standardization of message content is another key challenge for moving forward toward semantics in enterprise.

4. Results

The POC demonstrated a common and standard communications protocol and a standard and discoverable interface type. These permitted abstraction of the specifics and details of enterprise components, systems, utilities, and IT business processes. Each of these elements was able to communicate freely and discover each other using a common protocol. We were able to see true enterprise level autonomic behavior in which the low level objects and technologies behaved in a way that fit with the business needs to meet an IT service SLA, we also saw the business level involved in the autonomic control-loop to make the right decisions. Some of the more detailed results include: we were able to successfully interface existing manageability processes, services, toolsets, and the various managed objects with a reusable WS-Management protocol stack. The effort required to do this was minimal – more time was spent increasing the flexibility of the stack we were provided than in creating the functionality. We successfully demonstrated the use of the HTTP/SOAP protocol to do automated self-announcement of a device as it is introduced to the network. We used the same methodology to demonstrate self-announcement of adds or changes to software on the device. We integrated self-

announcement with automatic end-point interrogation over WS-Management; and successfully used WS-Management to automatically populate a CMDB with the data gathered through interrogation. We designed a means to define a grouping of behaviors under the umbrella of a policy and a way to identify system attributes to trigger the application of these policies. Using WS-Management, we communicated the need to apply the behaviors to a compliance system. We used WS-Eventing as a trigger to start an event management “transaction”. We made event management automatically intersect with ITIL processes, and communicate the need to apply a known solution to a compliance system over WS-Management protocol. We effectively demonstrated a form of well-behaved system self-management. We were able to automatically discover and provision new devices, just out-of-the-box, by having an Intel® AMT simulator initiate the discovery process out-of-band (OOB), then through policies provisioned the entire stack, resulting in addition of a new piece of hardware automatically to an existing IT service to provide additional computing power. We were also able to utilize WS-Catalog to represent and expose the MO metadata in a standard way to drive standards-based and free communications between components in a loosely-coupled manner.

5. Conclusions

The main conclusion is that web services technologies, as open standards, can dramatically simplify enterprise integration and promote the evolution of the autonomic enterprise. It can be shown that this end-state will result in lower TCO for IT shops and with better quality of service. Although still in progress, the POC work achieved a mature state in which the viability of the concept has been demonstrated. It clearly shows that open standards such as the WS-Management specification suite can significantly change the enterprise landscape, enabling autonomic response and resulting in a more agile IT required to respond to tomorrow’s business challenges. The POC shows that using today’s WS technologies as a standard messaging layer results in a common “language” throughout the manageability framework. But this result will not happen by itself. We found that the WS-Catalog is an important element for the ability to represent and expose the MO metadata in a standard way so components can freely communicate in a loosely-coupled manner. The criticality of a standard and discoverable manageability interface is evident, as well as the need to standardize the discovery process and include self-announcement in the WS-Management stack. We also found that Intel® AMT

plays a foundational role in platforms discovery and provisioning processes, as well as support for device autonomics. These enable a holistic approach to object management, but not by themselves alone. CMDB, for instance, has a central role in the enterprise and its functionality and data model need to be standardized. As a general observation, data standardization is needed in conjunction with the protocol standards being adopted currently. With this standardization, it will be possible to add new objects and services without needing to significantly alter the management framework. It will also be possible to replace management toolsets without needing to reconfigure the entire framework. Work needs to be done to further decouple the service end-points from the objects being managed. WS-Management is not a panacea. The protocol, and the concept this paper presents addresses a significant part of the manageability puzzle. During the POC, we found the need to enhance our existing event data model significantly to drive autonomics. Nevertheless we have proven that a major step toward such an end-state vision is possible.

This paper is for informational purposes only. THIS DOCUMENT IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE. Intel disclaims all liability, including liability for infringement of any proprietary rights, relating to use of information in this specification. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted herein.

Intel, the Intel logo, and Intel® Active Management Technology (Intel® AMT) are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

* Other names and brands may be claimed as the property of others.

Copyright © 2006, Intel Corporation. All rights reserved.